DERWENT-ACC-NO: 1992-422278

DERWENT-WEEK: 199251



TITLE: Accessing rows based on hidden field values - creating an index on bucket number hidden field and mapping to each row address in partitioned table

PATENT-ASSIGNEE: ANONYMOUS[ANON]

PRIORITY-DATA: 1992RD-0343006 (October 20, 1992)

PATENT-FAMILY:

| PUB-NO | PUB-DATE | LANGUAGE | PAGES | MAIN-IPC |
|---|---|---|---|---|
| **RD 343006** A | November 10, 1992 | N/A | 000 | G06F 000/00 |

APPLICATION-DATA:

| PUB-NO | APPL-DESCRIPTOR | APPL-NO | APPL-DATE |
|---|---|---|---|
| RD 343006A | N/A | 1992RD-0343006 | October 20, 1992 |

INT-CL (IPC): G06F000/00

ABSTRACTED-PUB-NO: RD 343006A

BASIC-ABSTRACT:

A hidden field containing the bucket number and a hidden field giving a row's address or ID (RID) can be exploited for accessing rows to be moved. An index on the bucket number (BNUM) hidden field is created and maintained. The bucket number refers to the result of hashing a partitioning key value in the rows of the table, determining which partition each row of the table will be placed in. There can be many more buckets than nodes, each node receiving the rows of several buckets. The index maps BNUM to RID for each row in the partitioned

table.

To access all the rows belonging to a given bucket or a given set of buckets,
the index is searched using the given bucket number(s) as search key(s). The
database management system's index manager is used for these scans. The RIDs
that are returned from the scan should be sorted, for the most efficient
retrieval. This can be done by using the database management system's sort
services in the normal manner, with the RID column being the sort key. The
sorted list of RIDs is then consulted, and each row pointed to by each RID is
retrieved, using the database management system's record retrieval service.

USE/ADVANTAGE - To reorganise or add nodes to a partitioned database. Good
performance, simplicity and concurrency. An index lookup prevents need to
access rows that are not in buckets of concern, and sorting RIDs allows
efficient retrieval of rows (clusters I/Os so same page need not be read more
than once). Uses existing services, i.e. a typical database management system
already has components that perform index lookups, sorts, and record retrieval.
No need to access and therefore lock rows that are not affected.

CHOSEN-DRAWING: Dwg.0/0

TITLE-TERMS: ACCESS ROW BASED HIDE FIELD VALUE INDEX BUCKET NUMBER HIDE FIELD
MAP ROW ADDRESS PARTITION TABLE

DERWENT-CLASS: T01

EPI-CODES: T01-J05B2;

SECONDARY-ACC-NO:
Non-CPI Secondary Accession Numbers: N1992-322101

| | Hits | Search Text | DBs |
|---|---|---|---|
| 1 | 0 | "creating an index on bucket".ti. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 2 | 1 | "RD 343006" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 3 | 0 | "92886751" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 4 | 65 | "5404510" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 5 | 64 | "5404510" | USPAT |
| 6 | 3 | "DATABASE MANAGING METHOD" | USPAT |
| 7 | 0 | 6 AND PARTITION$4 AND HASH$4 AND DATABASE$1 | USPAT |
| 8 | 0 | 6 AND PARTITION$4 AND HASH$4 | USPAT |
| 9 | 78 | DATABASE WITH MANAGING.TI. | USPAT |

| | Hits | Search Text | DBs |
|---|---|---|---|
| 10 | 0 | 9 AND partition$4 and (hash$4 with (value$1 or function$1)) and @ad < "20011016" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 11 | 3238 | DATABASE WITH MANAGING | USPAT |
| 12 | 79 | 11 AND partition$4 and (hash$4 with (value$1 or function$1)) and @ad < "20011016" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 13 | 76 | 12 AND DATABASE$1 AND (ROW$1 OR RECORD$1) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 14 | 19 | 13 and (partition$4 with key$1) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |
| 15 | 4 | 13 and (partition$4 adj key$1) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB |

09/981,613

File 347:JAPIO Dec 1976-2005/Dec(Updated 060404)
(c) 2006 JPO & JAPIO
File 350:Derwent WPIX 1963-2006/UD,UM &UP=200642
(c) 2006 The Thomson Corp.

| Set | Items | Description |
|---|---|---|
| S1 | 1617753 | DATABASE? ? OR DATABANK? ? OR DATA()(BASE? ? OR BANK? ?) OR DBMS OR RDBMS OR OODB OR STORAGE OR REPOSITOR??? OR TABLE? ? |
| S2 | 32326 | S1(3N)(PARTITION??? OR SECTION???) |
| S3 | 867858 | COLUMN? ? OR FIELD? ? |
| S4 | 128 | S3(3N)(HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUNCTION? ?) |
| S5 | 713 | PRIMARY(3N)(KEY? ? OR INDEX? ?) |
| S6 | 29341 | (ROW? ? OR TUPLE? ? OR RECORD? ?)(3N)(ORDER??? OR DISTRIBUT??? OR SORT??? OR ARRANG??? OR ARRANGEMENT? ?) |
| S7 | 1 | S2 AND S4 |
| S8 | 34 | S1 AND S4 |
| S9 | 1 | S8 AND S5 |
| S10 | 1 | S9 NOT S7 |
| S11 | 33 | S8 NOT S7 |
| S12 | 27 | S11 NOT AD=20011016:20031016/PR |
| S13 | 22 | S12 NOT AD=20031016:20061016/PR |
| S14 | 5 | S2 AND S5 |
| S15 | 5 | S14 NOT (S7 OR S10 OR S11) |
| S16 | 9 | S5(3N)(HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUNCTION? ?) |
| S17 | 8 | S16 NOT (S7 OR S10 OR S11) |
| S18 | 6 | S17 NOT AD=20011016:20031016/PR |
| S19 | 5 | S18 NOT AD=20031016:20061016/PR |
| S20 | 2 | S6 AND S4 |
| S21 | 37 | S6 AND (HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUNCTION? ?) |
| S22 | 52 | S2 AND (HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUNCTION? ?) |
| S23 | 17 | S22 AND KEY? ? |
| S24 | 53 | S21 OR S23 |
| S25 | 49 | S24 NOT (S7 OR S10 OR S11 OR S15 OR S17 OR S20) |
| S26 | 37 | S25 NOT AD=20011016:20031016/PR |
| S27 | 33 | S26 NOT AD=20031016:20061016/PR |
| S28 | 25 | S27 AND IC=G06F |

13/5/11 (Item 7 from file: 350)
DIALOG(R)File 350:Derwent WPIX


014191868 **Image available**
WPI Acc No: 2002-012565/200202
XRPX Acc No: N02-010358
**Star join operation method in relational** database **management system, involves selecting row and column from star map using hash row value and accessing fact** table **accordingly**
Patent Assignee: NCR INT INC (NATC ); NCR CORP (NATC )
Inventor: KOSTAMAA O P; RAMESH B; BHASHYAM R
Number of Countries: 027 Number of Patents: 002
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| EP 1148430 | A2 | 20011024 | EP 2001303147 | A | 20010402 | 200202 | B |
| US 6618729 | B1 | 20030909 | US 2000556009 | A | 20000420 | 200361 | |

Priority Applications (No Type Date): US 2000556009 A 20000420
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| EP 1148430 | A2 | E | 12 | G06F-017/30 | |
| | Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI TR | | | | |
| US 6618729 | B1 | | | G06F-017/00 | |

Abstract (Basic): EP 1148430 A2
NOVELTY - A cross-product is generated from dimension **tables** referenced by star join, and join **column** are **hashed** to create a hash-row value. A portion of hash row value is used to select a star map row and its another portion is used to select column of selected row. A fact **table** is accessed to join with cross-product when selected column indicates that the record exist in fact **table**.
DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:
(a) Computer implemented system for performing star join;
(b) Data structure;
(c) Computer program for performing star join
USE - For performing star join operation in relational **database** management system ( **RDBMS** ) of computer system such as mainframe, micro computer or personal computer system.
ADVANTAGE - Since hash-row value is used for addressing star map, the size of map is maintained constant, thus improves the performance of star joins.
DESCRIPTION OF DRAWING(S) - The figure shows a hardware and software environment for performing star join.
pp; 12 DwgNo 1/5
Title Terms: STAR; JOIN; OPERATE; METHOD; RELATED; **DATABASE** ; MANAGEMENT; SYSTEM; SELECT; ROW; COLUMN; STAR; MAP; HASH; ROW; VALUE; ACCESS; FACT; **TABLE** ; ACCORD
Derwent Class: T01
International Patent Class (Main): G06F-017/00; G06F-017/30
File Segment: EPI

13/5/20 (Item 16 from file: 350)
DIALOG(R)File 350:Derwent WPIX


010245854 **Image available**
WPI Acc No: 1995-147109/199519
XRPX Acc No: N95-115528

**Processor controlled index generation for relational** database **- by analysing workload of all requests in system, assigning value of importance to each request, breaking request into expressions, contexts and columns to ease identification of candidate indexes**

Patent Assignee: ORACLE CORP (ORAC-N)
Inventor: PANT S; SMITH G S

Number of Countries: 001 Number of Patents: 001
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| US 5404510 | A | 19950404 | US 92886751 | A | 19920521 | 199519 | B |

Priority Applications (No Type Date): US 92886751 A 19920521
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| US 5404510 | A | | 21 | G06F-015/40 | |


Abstract (Basic): US 5404510 A

The indexes are generated by the processor establishing indexes to provide efficient system operation. The processor looks at the logical scheme and workload of the **database** . The indexes are generated (48) by identifying importance values for the individual **database** requests. They are designed by identifying new indexes by order of request importance. Existing indexes are reused and modified based on the match between existing indexes (54). Previously identified indexes are searched for a similar index to each candidate index.

The importance value of the request is identified from expected frequency of processing and from individual importance of the request. Columns and associated operators for the individual **table** contexts are identified in each expression of each request. The candidate indexes are identified from the columns for individual contexts by order of request value of importance.

USE/ADVANTAGE - For SQL. Reduces time to locate records by using indexes, cheaper to scan indexes than **tables** , uses hash indexing by applying system **hash** to **column** data serving as key to index to locate page of index, more direct access to individual records.

Dwg.5/11

Title Terms: PROCESSOR; CONTROL; INDEX; GENERATE; RELATED; **DATABASE** ; ANALYSE; REQUEST; SYSTEM; ASSIGN; VALUE; IMPORTANT; REQUEST; BREAK; REQUEST; EXPRESS; CONTEXT; COLUMN; EASE; IDENTIFY; CANDIDATE; INDEX
Index Terms/Additional Words: PROCESSOR; CONTROL; INDEX; G
Derwent Class: T01
International Patent Class (Main): G06F-015/40
File Segment: EPI

**20/5/1 (Item 1 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


015340342 **Image available**
WPI Acc No: 2003-401280/200338
Related WPI Acc No: 2003-247532
XRPX Acc No: N03-320013

**Data file for organizing information in persistent computer storage device, has equally sized data frames into which records comprising key length/key data field and record length/data field are stored**

Patent Assignee: PRECISION SOLUTIONS INC (PREC-N)
Inventor: KING K D
Number of Countries: 001 Number of Patents: 001
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| US 6532476 | B1 | 20030311 | US 99438328 | A | 19991113 | 200338 | B |

Priority Applications (No Type Date): US 99438328 A 19991113
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| US 6532476 | B1 | | 60 | G06F-017/30 | |

Abstract (Basic): US 6532476 B1

NOVELTY - The data file has equally sized data frames into which records are stored. A file header comprises a frame size **field** , a **hash** type **field** and a modulo field which indicates the number of hash buckets to be used by the hashing algorithm indicated by the **hash** type **field** . The records have a key length field, a key data field, a record data field and a record length field.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) a method for storing records in persistent storage; and

(2) a method for accessing a record in data file stored in persistent storage.

USE - For organizing information in persistent computer storage device. As a multidimensional database.

ADVANTAGE - Allows record of unlimited dimensions containing data of any type and size, in any combination, to be constructed, maintained and utilized in the persistent storage. The hashing algorithm is used to balance the **distribution** of **records** into the data frames, thereby minimizing the access time to any one record.

DESCRIPTION OF DRAWING(S) - The figure shows a schematic view of a multi-dimensional data set.

pp; 60 DwgNo 1/39

Title Terms: DATA; FILE; ORGANISE; INFORMATION; PERSISTENT; COMPUTER; STORAGE; DEVICE; EQUAL; SIZE; DATA; FRAME; RECORD; COMPRISE; KEY; LENGTH; KEY; DATA; FIELD; RECORD; LENGTH; DATA; FIELD; STORAGE
Derwent Class: T01
International Patent Class (Main): G06F-017/30
File Segment: EPI

**28/5/2 (Item 2 from file: 347)**
DIALOG(R)File 347:JAPIO


06915216 **Image available**
DATABASE MANAGING METHOD

PUB. NO.: 2001-142752 [JP 2001142752 A]
PUBLISHED: May 25, 2001 (20010525)
INVENTOR(s): KAWAMURA NOBUO
HOSHINO RYUICHI
KASAO HIDEAKI
APPLICANT(s): HITACHI LTD
HITACHI SOFTWARE ENG CO LTD
APPL. NO.: 11-323657 [JP 99323657]
FILED: November 15, 1999 (19991115)
INTL CLASS: **G06F-012/00** ; **G06F-017/30**

ABSTRACT

PROBLEM TO BE SOLVED: To solve the problem that data need to be rearranged in an added database storage area when the database storage area is added as the database capacity increases.

SOLUTION: When (m) database storage areas are given as storage areas of a database, one or more data items of the **data base** are used as **partitioning keys** and the **database** is divided into (n) ($m \leq n$) packets as logical units by applying a **hash** function to the **partitioning keys** ; and the **database** is managed by using a **hash** map table which determines the correspondence of the data base storage areas managing the respective buckets according to the number of the given database storage areas and a segment **hash** map table for mapping distributed buckets with segments as storage units in the respective database storage areas.



**28/5/5 (Item 5 from file: 347)**
DIALOG(R)File 347:JAPIO


06665097 **Image available**
# METHOD AND SYSTEM FOR MANAGING DATABASE

PUB. NO.: 2000-250921 [JP 2000250921 A]
PUBLISHED: September 14, 2000 (20000914)
INVENTOR(s): GOSHO HIROYUKI
KIMURA KOJI
KONDO YOICHI
HARA KIYONOBU
TAKAZAWA KAZUYUKI
APPLICANT(s): HITACHI LTD
APPL. NO.: 11-049664 [JP 9949664]
FILED: February 26, 1999 (19990226)
INTL CLASS: **G06F-017/30** ; **G06F-012/00**

## ABSTRACT

PROBLEM TO BE SOLVED: To perform grouping operation which requires sorting by a grouping column value at high speed by providing a means for extracting records from a data base and grouping them by using a **hash** method and a means for **sorting** the grouped **records** .
SOLUTION: In **hash** grouping 102, ungrouped records are take out the data base, one by one, a **hash** value is obtained from a **hash** function 103 by using a grouping column value as a key, and grouped records are generated in a **hash** grouping area 104 and related in **sorting order** . In grouped **record** extraction 106, the grouped records are taken out of the **hash** grouping area 104 in the sorting order, which is always held. In **sort** grouping 107, ungrouped **records** stored in a work file 105 are sorted by the grouping column value to generate grouped records.

03594649 **Image available**

# **RECORD ARRANGEMENT** METHOD

PUB. NO.: 03-257549 [JP 3257549 A]
PUBLISHED: November 18, 1991 (19911118)
INVENTOR(s): CHIMURA YASUBUMI
APPLICANT(s): OKI ELECTRIC IND CO LTD [000029] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 02-054846 [JP 9054846]
FILED: March 08, 1990 (19900308)
INTL CLASS: [5] **G06F-012/00** ; **G06F-015/40**
JAPIO CLASS: 45.2 (INFORMATION PROCESSING -- Memory Units); 45.4 (INFORMATION PROCESSING -- Computer Applications)
JOURNAL: Section: P, Section No. 1312, Vol. 16, No. 60, Pg. 69, February 14, 1992 (19920214)

ABSTRACT

PURPOSE: To suppress the increase of the access frequency to an external storage and also to suppress the delay of the searching speed for a 1st file by setting the capacity of a unit information store area added with an index at the element number value larger than the maximum quantity of information that can be read into the external storage with a single access.

CONSTITUTION: A **hash** address (h) is obtained from a record key K, and a 2nd file VI is checked for reading an index (p) out of the address (h). Then the record stored in a 2nd file V is shecked based on the index (p). Thus both files V and VI are checked independently of each other and therefore at least two accesses are needed to an external storage. Under such conditions, the synonyms are stored contiguous to each other in the file V and the synonyms equal to the maximum quantity of information that can be read with a single access given to the external storage can be read en bloc. Thus it is possible to suppress the increase of the access frequency to the external storage and also to suppress the delay of the searching speed.

**28/5/12 (Item 12 from file: 347)**


02059533 **Image available**
JOINT PROCESSING SYSTEM FOR DECENTRALIZED DATA BASE

PUB. NO.: 61-273633 [JP 61273633 A]
PUBLISHED: December 03, 1986 (19861203)
INVENTOR(s): OKAZAKI TAKU
YAMANE YASUO
APPLICANT(s): FUJITSU LTD [000522] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 60-115824 [JP 85115824]
FILED: May 29, 1985 (19850529)
INTL CLASS: [4] **G06F-007/32** ; **G06F-012/00**
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.2 (INFORMATION PROCESSING -- Memory Units)
JOURNAL: Section: P, Section No. 571, Vol. 11, No. 134, Pg. 38, April 28, 1987 (19870428)

ABSTRACT

PURPOSE: To improve the processing efficiency in a joint process by transmitting the number of tuples at the side having the smaller number of **hash** sorts with the **hash** packets at both sites according to the result of the **hash** sorting.

CONSTITUTION: When a **hash** processing part 3 of a processor 1 is started, the tuples of the desired relation are read successively outpof a data base. These **tuples** are **sorted** to one of **hash** packets 11-0-11-p corresponding to the **hash** function valve of the prescribed join field value. then the number of tuples are informed to a short informing part 4. The part 4 transfers the number of tuples to a communication processing part 2 and an array comparison part 5. The part 5 receives the number of **hash** sorts from the site at the remote side and compares them with those at the own site. Then the part 5 transmits the tapples sorted to the packets even with the packet having a small number of **hash** sorts of the own site. The result of said comparison is sent to a joint processing part 6 for execution of the joint processing with the tuples received from the remote side for the packet whose joint processing should be applied at the own site.

**28/5/13 (Item 1 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


015513933 **Image available**
WPI Acc No: 2003-576080/200354
XRPX Acc No: N03-457915

**Records distribution determination method for computer system, involves locating marker data records in database and analyzing position of adjacent actual data records**

Patent Assignee: MICROSOFT CORP (MICT )
Inventor: BOA D S; MEACHAM S M; NOLTE B M
Number of Countries: 001 Number of Patents: 001
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| US 6560599 | B1 | 20030506 | US 99343625 | A | 19990630 | 200354 | B |

Priority Applications (No Type Date): US 99343625 A 19990630
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| US 6560599 | B1 | | 10 | G06F-017/30 | |

Abstract (Basic): US 6560599 B1

NOVELTY - The actual data records (1) and marker data record (7) which include pointer to logically preceding and proceeding actual data records, are inserted in a threaded linear **hash** table (4) using **hash** function. A known key of marker data record is **hashed** to locate marker data records in database and analyze position of adjacent actual data records using **hash** function for determining **distribution** of data **records**.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) **records distribution** determination system; and

(2) computer readable media storing **records distribution** determination program.

USE - For determining **distribution** of marker **records** in **hash** table used in various storage devices of computer system including personal computer and servers in local area network (LAN), wide area network (WAN) like Internet.

ADVANTAGE - The **distribution** of **records** in **hash** table is determined efficiently by locating marker data records in database and analyzing position of adjacent actual data **records** , thereby efficiently assessing **distribution** of **records** , optimizing quantity of data stored in memory and adjusting or tuning **hash** function to increase density of records around a given mark.

DESCRIPTION OF DRAWING(S) - The figure shows the **hash** table.

actual data records (1)

**hash** table (4)

marker data records (7)

pp; 10 DwgNo 2B/5

Title Terms: RECORD; DISTRIBUTE; DETERMINE; METHOD; COMPUTER; SYSTEM; LOCATE; MARK; DATA; RECORD; DATABASE; POSITION; ADJACENT; ACTUAL; DATA; RECORD
Derwent Class: T01
International Patent Class (Main): **G06F-017/30**
File Segment: EPI

**28/5/14 (Item 2 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


015136517 **Image available**
WPI Acc No: 2003-197043/200319
Related WPI Acc No: 2003-353655
XRPX Acc No: N03-156336

**Data record location method in computer system, involves inserting actual and marker data** records **having logical** ordering **specified by keys, in** hash **table using** hash **function**

Patent Assignee: MICROSOFT CORP (MICT )
Inventor: BOA D S; MEACHAM S M; NOLTE B M
Number of Countries: 001 Number of Patents: 001
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| US 6480858 | B1 | 20021112 | US 99345176 | A | 19990630 | 200319 | B |

Priority Applications (No Type Date): US 99345176 A 19990630
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| US 6480858 | B1 | | 11 | G06F-017/00 | |

Abstract (Basic): US 6480858 B1

NOVELTY - The actual and marker data **records** having a logical **ordering** specified by the keys, are inserted in a **hash** table using a **hash** function. The keys of marker data **records** are **distributed** at known positions throughout the range of keys of actual data records. One of the keys of the marker data records is **hashed** to locate the associated record in the **hash** table, if no record exists in the **hash** table for the input key.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for computer readable recorded medium storing data record locating program.

USE - For locating data record in computer system using **hash** table.

ADVANTAGE - Efficiently performs sequential linear access and other operations on logically ordered data stored in non-logical order in **hash** table.

DESCRIPTION OF DRAWING(S) - The figure shows the flowchart explaining data record locating process.

pp; 11 DwgNo 6/6

Title Terms: DATA; RECORD; LOCATE; METHOD; COMPUTER; SYSTEM; INSERT; ACTUAL ; MARK; DATA; RECORD; LOGIC; ORDER; SPECIFIED; KEY; **HASH** ; TABLE; **HASH** ; FUNCTION
Derwent Class: T01
International Patent Class (Main): **G06F-017/00**
File Segment: EPI

**28/5/16 (Item 4 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


012153796 **Image available**
WPI Acc No: 1998-570708/199849
XRPX Acc No: N98-444199

**Transforming records to be** hashed **stored in storage form processing system - uses group-by operation execution device for reading list of** hashed **records output to storage device by output device and** sorting hashed records **in list according to key value**

Patent Assignee: FUJITSU LTD (FUIT )
Inventor: AKABOSHI N; HARADA L; OGIHARA K; TAKE R
Number of Countries: 026 Number of Patents: 006

Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| EP 877324 | A2 | 19981111 | EP 98302400 | A | 19980327 | 199849 | B |
| JP 11003342 | A | 19990106 | JP 97152171 | A | 19970610 | 199911 | |
| US 6226634 | B1 | 20010501 | US 9849091 | A | 19980327 | 200126 | |
| JP 3466054 | B2 | 20031110 | JP 97152171 | A | 19970610 | 200377 | |
| JP 2004062898 | A | 20040226 | JP 97152171 | A | 19970610 | 200416 | |
| | | | JP 2003196820 | A | 20030714 | | |
| JP 3601719 | B2 | 20041215 | JP 97152171 | A | 19970610 | 200482 | |
| | | | JP 2003196820 | A | 20030714 | | |

Priority Applications (No Type Date): JP 97152171 A 19970610; JP 97100696 A 19970418

Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| EP 877324 | A2 | E | 215 | G06F-017/30 | |
| Designated States (Regional): AL AT BE CH DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI | | | | | |
| JP 11003342 | A | | 98 | G06F-017/30 | |
| US 6226634 | B1 | | | G06F-017/00 | |
| JP 3466054 | B2 | | 92 | G06F-017/30 | Previous Publ. patent JP 11003342 |
| JP 2004062898 | A | | 102 | G06F-017/30 | Div ex application JP 97152171 |
| JP 3601719 | B2 | | 102 | G06F-017/30 | Div ex application JP 97152171<br>Previous Publ. patent JP 2004062898 |

Abstract (Basic): EP 877324 A

The system includes a record storing device (106) for temporarily storing the records. A pointer storing device (107) is used for storing pointers to the records in the record storing device (106) at positions. Each of the latter corresponds to a **hash** function value calculated using the key value of the pointed record. An output device (108) outputs the records pointed to by the pointers stored in the pointer storing device (107) to the storage device, given the **hash** function values for storage positions of the pointers.

A group-by operation execution device (109) is used for reading a list of **hashed** records output to the storage device by the output device (108), **sorting** the **hashed records** in the list according to the key value, and performing the group-by operation on the list of the **sorted records** .

USE - For processing large amount of data stored in database for obtaining rule of relationship among data stored in database.:

ADVANTAGE - Provides high speed using result of **hash** process.

Dwg.1a/171

Title Terms: TRANSFORM; RECORD; **HASH** ; STORAGE; STORAGE; FORM; PROCESS; SYSTEM; GROUP; OPERATE; EXECUTE; DEVICE; READ; LIST; **HASH** ; RECORD; OUTPUT; STORAGE; DEVICE; OUTPUT; DEVICE; SORT; **HASH** ; RECORD; LIST; ACCORD; KEY; VALUE

Derwent Class: T01
International Patent Class (Main): **G06F-017/00** ; **G06F-017/30**
International Patent Class (Additional): **G06F-007/24** ; **G06F-012/00** ; **G06F-019/00**
File Segment: EPI

**28/5/22 (Item 10 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


009294868
WPI Acc No: 1992-422278/199251
XRPX Acc No: N92-322101
**Accessing rows based on hidden field values - creating an index on bucket number hidden field and mapping to each row address in** partitioned table
Patent Assignee: ANONYMOUS (ANON )
Number of Countries: 001 Number of Patents: 001
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| RD 343006 | A | 19921110 | RD 92343006 | A | 19921020 | 199251 | B |

Priority Applications (No Type Date): RD 92343006 A 19921020
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| RD 343006 | A | | | G06F-000/00 | |

Abstract (Basic): RD 343006 A

A hidden field containing the bucket number and a hidden field giving a row's address or ID (RID) can be exploited for accessing rows to be moved. An index on the bucket number (BNUM) hidden field is created and maintained. The bucket number refers to the result of **hashing** a partitioning **key** value in the rows of the **table** , determining which **partition** each row of the table will be placed in. There can be many more buckets than nodes, each node receiving the rows of several buckets. The index maps BNUM to RID for each row in the **partitioned table** .

To access all the rows belonging to a given bucket or a given set of buckets, the index is searched using the given bucket number(s) as search **key** (s). The database management system's index manager is used for these scans. The RIDs that are returned from the scan should be sorted, for the most efficient retrieval. This can be done by using the database management system's sort services in the normal manner, with the RID column being the sort **key** . The sorted list of RIDs is then consulted, and each row pointed to by each RID is retrieved, using the database management system's record retrieval service.

USE/ADVANTAGE - To reorganise or add nodes to a **partitioned database** . Good performance, simplicity and concurrency. An index lookup prevents need to access rows that are not in buckets of concern, and sorting RIDs allows efficient retrieval of rows (clusters I/Os so same page need not be read more than once). Uses existing services, i.e. a typical database management system already has components that perform index lookups, **sorts** , and **record** retrieval. No need to access and therefore lock rows that are not affected.

Dwg.0/0
Title Terms: ACCESS; ROW; BASED; HIDE; FIELD; VALUE; INDEX; BUCKET; NUMBER; HIDE; FIELD; MAP; ROW; ADDRESS; PARTITION; TABLE
Derwent Class: T01
International Patent Class (Main): **G06F-000/00**
File Segment: EPI

**28/5/24 (Item 12 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


008122950 **Image available**
WPI Acc No: 1990-009951/199002
XRPX Acc No: N90-007625

**Real-time database for computer integrated mfg. system - stores, searches, and retrieves tuples in data tables, and stores and retrieves informatted data in input areas**

Patent Assignee: HEWLETT-PACKARD CO (HEWP )
Inventor: FATEHI F; GIVENS C; HONG L T; LIU C; WRIGHT M J; LUI C; LUI C C
Number of Countries: 005 Number of Patents: 005
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| EP 350208 | A | 19900110 | EP 89306620 | A | 19890629 | 199002 | B |
| US 4961139 | A | 19901002 | US 88213578 | A | 19880630 | 199042 | |
| CA 1319756 | C | 19930629 | CA 604425 | A | 19890629 | 199332 | |
| EP 350208 | B1 | 19970108 | EP 89306620 | A | 19890629 | 199707 | |
| DE 68927621 | E | 19970220 | DE 627621 | A | 19890629 | 199713 | |
| | | | EP 89306620 | A | 19890629 | | |

Priority Applications (No Type Date): US 88213578 A 19880630
Cited Patents: 4.Jnl.Ref; A3...9148; No-SR.Pub
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| EP 350208 | A | E | 16 | | |
| EP 350208 | B1 | E | 20 | G06F-017/30 | |
| Designated States (Regional): DE FR GB | | | | | |
| DE 68927621 | E | | | G06F-017/30 | Based on patent EP 350208 |
| CA 1319756 | C | | | G06F-015/40 | |

Abstract (Basic): EP 350208 A

A real-time database comprises data storage routines, data retrieval routines, data updating routines, and an index **hashing** mechanism, for storing, searching and retrieving tuples in data tables, and for storing and retrieving unformatted data in input areas. The data retrieval routines include a routine include a routine to directly access to data using tuple identifiers, and a routine to directly access unformatted data from input areas. The data retrieval routines for accessing tuples in data tables include an option to read-through-lock to access tuples in locked data tables.

The data updating routines include an option to omit index updating when updating data and an option to update data in a locked data table. The data storage routines and the data retrieval routines are independent and use has index tables to relatee an index key to an entry in the data table, so that multiple indexes can be defined for a data table. The data table structure includes a column defined for storing tuple identifier strings.

ADVANTAGE - Provides high speed data access required for on-line applications.

1/5

Title Terms: REAL; TIME; DATABASE; COMPUTER; INTEGRATE; MANUFACTURE; SYSTEM ; STORAGE; SEARCH; RETRIEVAL; DATA; TABLE; STORAGE; RETRIEVAL; DATA; INPUT; AREA
Derwent Class: T01
International Patent Class (Main): **G06F-015/40 ; G06F-017/30**
International Patent Class (Additional): **G06F-012/00**
File Segment: EPI

**28/5/25 (Item 13 from file: 350)**
DIALOG(R)File 350:Derwent WPIX


003580495
WPI Acc No: 1983-C8691K/198309
XRPX Acc No: N83-037329

**Storage sub-system with cache addressing - accommodates both discontinuous backing storage and preferential data segments while minimising clashing**

Patent Assignee: IBM CORP (IBMC )
Inventor: BENHASE M T; DUKE A H
Number of Countries: 013 Number of Patents: 007
Patent Family:

| Patent No | Kind | Date | Applicat No | Kind | Date | Week | |
|---|---|---|---|---|---|---|---|
| EP 72413 | A | 19830223 | EP 82105765 | A | 19820629 | 198309 | B |
| AU 8286111 | A | 19830224 | | | | 198314 | |
| ES 8305963 | A | 19830716 | | | | 198339 | |
| US 4464713 | A | 19840807 | US 81293648 | A | 19810817 | 198434 | |
| CA 1180463 | A | 19850102 | | | | 198506 | |
| EP 72413 | B | 19880504 | | | | 198818 | |
| DE 3278444 | G | 19880609 | | | | 198824 | |

Priority Applications (No Type Date): US 81293648 A 19810817
Cited Patents: 1.Jnl.Ref; A3...8504; EP 19358; GB 2052118; No-SR.Pub; US 4056845; US 4195342; US 4215402; EP 66766
Patent Details:

| Patent No | Kind | Lan | Pg | Main IPC | Filing Notes |
|---|---|---|---|---|---|
| EP 72413 | A | E | 38 | | |
| Designated States (Regional): BE CH DE FR GB IT LI NL SE | | | | | |
| EP 72413 | B | E | | | |
| Designated States (Regional): BE CH DE FR GB IT LI NL SE | | | | | |

Abstract (Basic): EP 72413 A

A controller monitors the controls of a cache containing copies of a variable subset of the records in backing storage together with a directory for locating the same, and for observing a **hashing** protocol in a **hash** mechanism. The backing storage is modular and has functional discontinuity between modules, each module having a multiple record capacity, with the expectancy of certain known preferred modules being more frequently accessed than other modules.

The directory is maintained in the form of independent strings of entries incorporating link elements and a table is maintained mapping the classes defined under the **hashing** protocol onto the directory strings. The **hashing** mechanism accesses the table to access the entries of a string sequentially in link order to a match or to the end of the string. The **hashing** mechanism maps records onto table entries such that, given a table entry **order** , sequential **records** within a module map onto sequential table entries, and no record in a preferred module maps onto the same table entry as does any record on any other preferred module.

Title Terms: STORAGE; SUB; SYSTEM; CACHE; ADDRESS; ACCOMMODATE; DISCONTINUE ; BACKING; STORAGE; PREFER; DATA; SEGMENT; MINIMISE; CLASH
Derwent Class: T01
International Patent Class (Additional): **G06F-001/00** ; **G06F-009/06** ; **G06F-012/10** ; **G06F-013/00** ; G11C-009/06
File Segment: EPI

```
File   2:INSPEC 1898-2006/Jun W4
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Jun W4
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Jun W4
         (c) 2006 Elsevier Eng.  Info. Inc.
File  23:CSA Technology Research Database 1963-2006/Jun
         (c) 2006 CSA.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Jun W4
         (c) 2006 Inst for Sci Info
File  35:Dissertation Abs Online 1861-2006/Jun
         (c) 2006 ProQuest Info&Learning
File  65:Inside Conferences 1993-2006/Jul 05
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Apr W1
         (c)2006 Japan Science and Tech Corp(JST)
File  95:TEME-Technology & Management 1989-2006/Jun W4
         (c) 2006 FIZ TECHNIK
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/May
         (c) 2006 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2006/Jun 22
         (c) 2006 The Gale Group
File 144:Pascal 1973-2006/Jun W2
         (c) 2006 INIST/CNRS
File 239:Mathsci 1940-2006/Aug
         (c) 2006 American Mathematical Society
File 256:TecInfoSource 82-2006/Aug
         (c) 2006 Info.Sources Inc
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info

Set     Items   Description
S1    2720371   DATABASE? ? OR DATABANK? ? OR DATA()(BASE? ? OR BANK? ?) OR
              DBMS OR RDBMS OR OODB OR STORAGE OR REPOSITOR??? OR TABLE? ?
S2       9041   S1(3N)(PARTITION??? OR SECTION???)
S3    6862222   COLUMN? ? OR FIELD? ?
S4       1204   S3(3N)(HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUN-
             CTION? ?)
S5       3258   PRIMARY(3N)(KEY? ? OR INDEX? ?)
S6      19360   (ROW? ? OR TUPLE? ? OR RECORD? ?)(3N)(ORDER??? OR DISTRIBU-
             T??? OR SORT??? OR ARRANG??? OR ARRANGEMENT? ?)
S7          1   S2 AND S4
S8          1   S2 AND S5
S9        105   S2 AND (HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FU-
             NCTION? ?)
S10         4   S9 AND S6
S11         1   RD  (unique items)
S12        77   S1 AND S4
S13         0   S12 AND S5
S14         0   S12 AND S6
S15         0   S14 AND PARTITION???
S16         5   S9 AND KEY? ?
S17         5   S16 NOT (S7 OR S8 OR S10)
S18         5   RD  (unique items)
S19        11   S9 AND (ROW? ? OR TUPLE? ? OR RECORD? ?)
S20         7   RD  (unique items)
S21         6   S20 NOT S17
S22        23   S5(3N)(HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FUN-
             CTION? ?)
S23        10   RD  (unique items)
S24        10   S23 NOT (S7 OR S8 OR S10 OR S17 OR S21)
```

```
S25         10   S24 NOT PY=2002:2006
S26         25   S2 AND S6
S27         13   RD  (unique items)
S28         12   S27 NOT (S7 OR S8 OR S10 OR S17 OR S21 OR S24)
S29         11   S28 NOT PY=2002:2006
S30        170   S6 AND (HASH??? OR DIGEST??? OR (ONE()WAY OR ONEWAY)(2N)FU-
                 NCTION? ?)
S31         93   S30 AND S1
S32         15   S31 AND (PARTITION??? OR GROUP???)
S33         10   RD  (unique items)
S34          9   S33 NOT (S7 OR S8 OR S10 OR S17 OR S21 OR S24)
S35          6   S34 NOT PY=2002:2006
```

7/5/1 **(Item 1 from file: 35)**


01526267 ORDER NO: AAD13-81133

**JOIN ALGORITHMS FOR PARALLEL COMPUTERS FOR RELATIONS BASED ON INTERPOLATION BASED GRID FILE**

Author: MOHAMMED, SALAHADIN ADEM
Degree: M.SC.
Year: 1991
Corporate Source/Institution: KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS (SAUDI ARABIA) (1088)
Source: VOLUME 35/01 of MASTERS ABSTRACTS.
PAGE 267. 167 PAGES
Descriptors: COMPUTER SCIENCE
Descriptor Codes: 0984

The recent advances in parallel and distributed processing and its applications to database operations such as join have initiated extensive research in this **field** . Investigations on **hash** based join algorithms compared to methods such as join-index join and merge-sort have given encouraging results. However, they involve a costly data partitioning phase prior to the join. This costly partitioning phase can be avoided if file structures that keep data already **partitioned** in the secondary **storage** are used. Interpolation Based Grid File (IBGF) is such a file structure. In this thesis new join algorithms for parallel computers for relations based on IBGF are investigated. Different algorithms are used for uniform relations and nonuniform relations. The efficiencies of these algorithms based on relations and architecture, have been studied using simulation. However the comparison of different techniques is not within the scope of this work.

**11/5/1 (Item 1 from file: 2)**
DIALOG(R)File 2:INSPEC


06159753 INSPEC Abstract Number: C9602-6160B-018
**Title: Dynamic load balancing in multicomputer database systems using partition tuning**
Author(s): Hua, K.A.; Chiang Lee; Hua, C.M.
Author Affiliation: Dept. of Comput. Sci., Univ. of Central Florida, Orlando, FL, USA
Journal: IEEE Transactions on Knowledge and Data Engineering vol.7, no.6 p.968-83
Publisher: IEEE,
Publication Date: Dec. 1995 Country of Publication: USA
ISSN: 1041-4347
SICI: 1041-4347(199512)7:6L.968:DLBM;1-#
Material Identity Number: N571-96001
U.S. Copyright Clearance Center Code: 1041-4347/95/$4.00
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P); Theoretical (T)
Abstract: Shared nothing multiprocessor architecture is known to be more scalable to support very large databases. Compared to other join strategies, a hash-based join algorithm is particularly efficient and easily parallelized for this computation model. However, this hardware structure is very sensitive to the skew in tuple distribution. Unless the parallel hash join algorithm includes some dynamic load balancing mechanism, the skew effect can severely deteriorate the system performance. In this paper, we investigate this issue. In particular, three parallel hash join algorithms are presented. We implement a simulator to study the effectiveness of these schemes. The simulation model is validated by comparing the simulation results to those produced by the actual implementation of the algorithms running on a multiprocessor system. Our performance study indicates that a naive approach is not able to provide tangible savings. However, the carefully designed strategies can offer substantial improvement over conventional techniques for a wide range of skew conditions. (35 Refs)
Subfile: C
Descriptors: database machines; database theory; distributed databases; parallel algorithms; query processing; relational databases; resource allocation; software performance evaluation; very large databases
Identifiers: dynamic load balancing; multicomputer database; partition tuning; shared nothing multiprocessor architecture; scalable architecture; very large databases; hash-based join algorithm; computation model; skew; tuple distribution; parallel hash join algorithm; system performance; parallel **hash** join algorithms; simulation model; multiprocessor system; performance study; database machine; relational database
Class Codes: C6160B (Distributed databases); C6150J (Operating systems); C4250 (Database theory); C6160D (Relational databases); C6150N ( Distributed systems software)

**25/5/3 (Item 3 from file: 2)**
DIALOG(R)File 2:INSPEC


05004382 INSPEC Abstract Number: C91072145
**Title: Trie hashing with controlled load**
Author(s): Litwin, W.A.; Roussopoulos, N.; Levy, G.; Hong, W.
Author Affiliation: Dept. of Comput. Sci., Maryland Univ., College Park, MD, USA
Journal: IEEE Transactions on Software Engineering vol.17, no.7 p. 678-91
Publication Date: July 1991 Country of Publication: USA
CODEN: IESEDJ ISSN: 0098-5589
U.S. Copyright Clearance Center Code: 0098-5589/91/0700-0678$01.00
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Trie **hashing** (TH), a **primary key** access method for storing and accessing records of dynamic files, is discussed. The key address is computed through a trie. A key search usually requires only one disk access when the trie is in core and two disk accesses for very large files when the trie must be on disk. A refinement to trie hashing, trie hashing with controlled load (THCL), is presented. It is designed to control the load factor of a TH file as tightly as that of a B-tree file, allows high load factor of up to 100% for ordered insertions, and increases the load factor for random insertions from 70% to over 85%. It is shown that these properties make trie hashing preferable to a B-tree. (29 Refs)
Subfile: C
Descriptors: file organisation; information retrieval systems; trees (mathematics)
Identifiers: primary key access method; dynamic files; key search; disk access; trie hashing; controlled load; THCL; load factor; TH file; B-tree file; high load factor; ordered insertions; load factor; random insertions
Class Codes: C6120 (File organisation); C7250 (Information storage and retrieval); C1160 (Combinatorial mathematics)

**25/5/8 (Item 8 from file: 2)**
DIALOG(R)File 2:INSPEC


02615393 INSPEC Abstract Number: C81002459
**Title: Automatic generation of storage structures for a DBTG data base system**
Author(s): Blanken, H.M.
Author Affiliation: Twente Univ. of Technol., Enschede, Netherlands
Conference Title: Proceedings of the International Conference on Data Bases p.99-118
Editor(s): Deen, S.M.; Hammersley, P.
Publisher: Heyden & Son, London, UK
Publication Date: 1980 Country of Publication: UK xii+288 pp.
ISBN: 0 85501 495 4
Conference Date: 2-4 July 1980 Conference Location: Aberdeen, UK
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: An aid is described which can assist the data base administrator in selecting optimally the storage structures for a data base. This data base has a network structure according to the CODASYL specifications. It is assumed that the administrator can define record and set types. It is also assumed that the data base system offers the following options, which may be selected by the data base administrator: an index can be defined on each attribute of a record type; non-singular sets can be implemented by means of pointer-arrays which are stored either adjacent to the owner record or are connected to this record by means of a pointer; and **hashing** on the **primary key** can be used to store and retrieve records and if hashing is not used then the clustering of those records on attribute value is possible. In order to select the storage structures a data base specification and load definition language is needed. The chosen object function is the minimization of the number of page accesses. (9 Refs)
Subfile: C
Descriptors: data structures; database management systems; minimisation
Identifiers: DBTG data base system; data base administrator; storage structures; network structure; CODASYL specifications; attribute; records; hashing; load definition language; object function; minimization; page accesses; non singular sets; pointer arrays; data structures; DBMS
Class Codes: C6120 (File organisation); C6160 (Database management systems (DBMS))

**25/5/7 (Item 7 from file: 2)**
DIALOG(R)File 2:INSPEC


02948812 INSPEC Abstract Number: C82043052
**Title: Partial-match retrieval for dynamic files**
Author(s): Lloyd, J.W.; Ramamohanarao, K.
Author Affiliation: Dept. of Computer Sci., Univ. of Melbourne, Melbourne, Vic., Australia
Journal: BIT (Nordisk Tidskrift for Informationsbehandling) vol.22, no.2 p.150-68
Publication Date: 1982 Country of Publication: Sweden
CODEN: NBITAB ISSN: 0006-3835
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Studies file designs for answering partial-match queries for dynamic files. A partial-match query is a specification of the value of zero or more fields in a record. An answer to query consists of a listing of all records in the file satisfying the values specified. The main contribution is a general method whereby certain **primary key hashing** schemes can be extended to partial-match retrieval schemes. These partial-match retrieval designs can handle arbitrarily dynamic files and can be optimized with respect to the number of page faults required to answer a query. The authors illustrate the method by considering in detail the extension of two recent dynamic **primary key hashing** schemes. (11 Refs)
Subfile: C
Descriptors: information retrieval; table lookup
Identifiers: information retrieval; table lookup; dynamic files; file designs; partial-match queries; **primary key hashing** schemes
Class Codes: C6120 (File organisation); C7250 (Information storage and retrieval)

29/5/5 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC


04982195 INSPEC Abstract Number: C91064868
**Title: Combined optimal** tuple ordering **and attribute** partitioning **in** storage **schema design**
Author(s): Gorla, N.; Quinn, W.
Author Affiliation: Dept. of Comput. & Inf. Sci., Cleveland State Univ., OH, USA
Journal: Information and Software Technology vol.33, no.5 p.335-9
Publication Date: June 1991 Country of Publication: UK
CODEN: ISOTE7 ISSN: 0950-5849
U.S. Copyright Clearance Center Code: 0950-5849/91/050335-05
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Database operating efficiency in relational databases can be improved significantly by using proper storage schema. Two such methods to improve performance are attribute partitioning and record clustering. In the past, the two design problems were treated separately-the optimal attribute partitioning was obtained without considering the **order** in which **tuples** are laid out; this does not produce a true optimal solution. The paper provides experimental evidence that the best attribute partitioning scheme is dependent on the **tuple ordering** and vice versa. The combined optimal solution of attribute partitioning and **tuple ordering** obtained in the experiments showed an improvement in database operating efficiency ranging from 12% to 35%, compared to the independent optimal solutions of attribute partitioning and **tuple ordering** considered separately. (6 Refs)
Subfile: C
Descriptors: relational databases; software engineering
Identifiers: optimal **tuple ordering** ; attribute partitioning; storage schema design; relational databases; record clustering; database operating efficiency
Class Codes: C6160D (Relational DBMS); C6110B (Software engineering techniques)

**35/5/1 (Item 1 from file: 2)**
DIALOG(R)File 2:INSPEC


07824730 INSPEC Abstract Number: C2001-03-7102-005
**Title: Exploiting early sorting and early** partitioning **for decision support query processing**
Author(s): Claussen, J.; Kemper, A.; Kossmann, D.; Wiesner, C.
Author Affiliation: Lehrstuhl fur Inf., Passau Univ., Germany
Journal: VLDB Journal vol.9, no.3 p.190-213
Publisher: Springer-Verlag,
Publication Date: Dec. 2000 Country of Publication: Germany
CODEN: VLDBFR ISSN: 1066-8888
SICI: 1066-8888(200012)9:3L.190:EESE;1-7
Material Identity Number: O851-2001-001
U.S. Copyright Clearance Center Code: 1066-8888/2000/$2.00+0.20
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Decision support queries typically involve several joins, a **grouping** with aggregation, and/or **sorting** of the result **tuples** . We propose two new classes of query evaluation algorithms that can be used to speed up the execution of such queries. The algorithms are based on: (1) early sorting and (2) early **partitioning** , or a combination of both. The idea is to push the sorting and/or the **partitioning** to the leaves, i.e., the base relations of the query evaluation plans (QEPs) and thereby avoid sorting or **partitioning** large intermediate results generated by the joins. Both early sorting and early **partitioning** are used in combination with **hash** based algorithms for evaluating the join(s) and the **grouping** . To enable early sorting, the sort order generated at an early stage of the QEP is retained through an arbitrary number of order-preserving **hash** joins. To make early **partitioning** applicable to a large class of decision support queries, we generalize the so-called **hash** teams proposed by G. Graefe et al. (1998). **Hash** teams allow one to perform several **hash** based operations (join and **grouping** ) on the same attribute in one pass without repartitioning intermediate results. Our generalization consists of indirectly **partitioning** the input data. Indirect **partitioning** means **partitioning** the input data on an attribute that is not directly needed for the next **hash** -based operation, and it involves the construction of bitmaps to approximate the **partitioning** for the attribute that is needed in the next **hash** based operation. Our performance experiments show that such QEPs based on early sorting, early **partitioning** , or both, perform significantly better than conventional strategies for many common classes of decision support queries. (33 Refs)
Subfile: C
Descriptors: decision support systems; query processing; relational algebra; sorting; **storage** management; very large **databases**
Identifiers: early sorting; early **partitioning** ; decision support query processing; joins; aggregation; result tuples; query evaluation algorithms; base relations; query evaluation plans; **hash** based algorithms; sort order ; QEP; order-preserving **hash** joins; **hash** teams; **hash** based operations ; intermediate results; indirect **partitioning** ; bitmaps
Class Codes: C7102 (Decision support systems); C6160Z (Other DBMS); C4250 (Database theory); C6130 (Data handling techniques); C6160D ( Relational databases); C4210 (Formal logic); C6120 (File organisation)